1. (Currently Amended) A machine-readable medium having stored thereon sequences of instructions which, when executed by a processor, cause the processor to perform the acts of:

disabling access to at least a first section of computer code in a network driver software interface that is being executed by the processor by overwriting computer code that is executed before the first section of computer code with blocking computer code, wherein the network driver software interface provides for communication between one or more media access control units and one or more protocol drivers in a computer system according to a set of bindings;

executing the blocking computer code with the processor;

patching the first section of computer code while the blocking computer code of the network driver software interface is being executed by the processor and without stopping complete operation of the network driver software interface, the patching of the first section of code comprising inserting a template jump for forwarding call instructions from the network driver software interface to a template in a rerouting driver, the template comprising new computer code for controlling communications, ~~in order to cause the insertion of a~~ the template jump and template allowing the rerouting driver to control communication between one or more media access control units and one or more protocol drivers in the computer system ~~into the one or more communication paths provided by the set of bindings~~; and

re-enabling access to the patched first section of computer code by replacing the blocking computer code with ~~computer code that allows execution of~~ the patched first section of computer code.

2. (Original) The machine-readable medium of claim 1 wherein the patching is static patching.

3. (Cancelled).

4. (Previously Presented) The machine-readable medium of claim 2, wherein template jumps are inserted in the network driver software interface so that a CALL instruction to the protocol driver is replaced with a JUMP to the template in the rerouting driver, the template containing the CALL instruction.

5. (Previously Presented) The machine-readable medium of claim 2 wherein the patching of the first section of computer code creates at least one new binding between the network driver software interface and the rerouting driver.

6. (Original) The machine-readable medium of claim 5 wherein the at least one new binding provides for communication between one or more media access control units and a capturing unit in the rerouting driver.

7. (Original) The machine-readable medium of claim 6 wherein the capturing unit is used to intercept communications over the at least one new binding.

8. (Original) The machine-readable medium of claim 1 wherein the patching is dynamic patching.

9. (Previously Presented) The machine-readable medium of claim 8 wherein the dynamic patching includes establishing a new binding between at least one media access control unit and dynamic patching computer code in the rerouting driver.

10. (Previously Presented) The machine-readable medium of claim 9 wherein template jumps are inserted in the network driver software interface so that a CALL instruction to the protocol driver is replaced with a JUMP to the template in the rerouting driver, the template containing the CALL instruction.

11. (Currently Amended) A computer implemented method comprising:

transmitting from a remote host to a first target computer on a network an installation application and a rerouting driver;

transmitting from the remote host to the first target computer a command to cause the first target computer to execute the installation application;

the first target computer, responsive to receipt of the command, executing the installation application, wherein the first target computer includes a network driver software interface that provides for communication between one or more media access control units and one or more protocol drivers according to a set of bindings; and

the first target computer, responsive to executing the installation application, causing the modification of the network driver software interface to insert the rerouting driver into the one or more communication paths provided by the set of bindings while the network driver software interface is being executed by the first target computer and without restarting the first target computer, the first target computer comprising a multiprocessor system, wherein the insert of the rerouting driver, further comprises:

the installation application disabling access to a least a first section of code in the network driver software interface by overwriting code prior to the first section with blocking code and without stopping complete operation of the network driver software interface;

the installation application patching the first section of code while the blocking code is being executed by the processor, the patching comprising inserting a template jump for forwarding call instructions from the network driver software interface to a template in the rerouting driver, the template comprising new computer code for controlling communications, the template jump and template allowing the rerouting driver to control communication between the one or more media access control units and the one or more protocol drivers.

12. (Previously Presented) The computer implemented method of claim 11 wherein the modification of the network driver software interface is by static patching.

13. (Cancelled).

14. (Previously Presented) The computer implemented method of claim 12 wherein the template jumps are inserted in the network driver software interface so that a CALL instruction to the protocol driver is replaced with a JUMP to the template in the rerouting driver, the template containing the CALL instruction.

15. (Original) The computer implemented method of claim 11 wherein the modification of the network driver interface is by dynamic patching.

16. (Previously Presented) The computer implemented method of claim 15 wherein the dynamic patching further comprises establishing a new binding between at least one media access control unit and dynamic patching code in the rerouting driver.

17. (Previously Presented) The computer implemented method of claim 16 wherein the template jumps are inserted in the network driver software interface so that a CALL instruction to the protocol driver is replaced with a JUMP to the template in the rerouting driver, the template containing the CALL instruction.

**[The Remainder of this Page Has Been Intentionally Left Blank]**

18. (Currently Amended) A computer system comprising:

a processor for simultaneously executing:

a protocol driver;

a network driver software interface;

a media access control unit; and

a rerouting driver, wherein during installation of the rerouting driver, a first section of code in the network driver software interface is disabled by overwriting code that is positioned before the first section of code with blocking code so that complete operation of the network driver software interface is not stopped, and wherein the first section of code is then patched by inserting a template jump for forwarding call instructions from the network driver software interface to a template in the rerouting driver, the template comprising new computer code for controlling communications, the template jump and template allowing the rerouting driver to control communication between the one or more media access control units and the one or more protocol drivers;

the network driver software interface to store a first binding defining a communication path between the protocol driver and the media access control unit, the network driver software interface coupled to communicate packets with the media access control unit, the network driver software interface being patched to communicate the packets with the rerouting driver; and

the rerouting driver being executed by the processor at the same time as the network driver software interface and being coupled to communicate the packets with the protocol driver.

19. (Original) The computer system of claim 18, the rerouting driver further comprising static patching code.

20. (Original) The computer system of claim 18, the rerouting driver further comprising dynamic patching code.

21. (Original) The computer system of claim 18, the rerouting driver further comprising a capture unit to store in a buffer one or more of the packets for evaluation

22. (Previously Presented) The computer system of claim 21, the network interface also stores a second binding defining a communication path between the rerouting driver and the media access control unit; and, the capture unit to store in the buffer the packets destined for the rerouting driver.

**[The Remainder of this page has been left intentionally blank.]**

23. (Currently Amended) A rerouting driver for remotely installing network drivers and software in a computer system without restarting the computer system following installation, the computer system having an operating system and multiple processors in which a network driver software interface provides communication of information between at least one media access control unit and at least one protocol driver on the computer system, the rerouting driver comprising:

control code, for controlling the rerouting driver;

binding code, for establishing at least one binding at the network driver software interface so that the rerouting driver is bound to at least one media access control unit while the network driver software interface and the rerouting driver are executed at the same time;

patching code, for inserting template jumps into at least a first section of code in the network driver software interface, the template jumps providing jumps to templates in the rerouting driver so that information from at least one media access control unit destined for at least one protocol driver is rerouted to the rerouting driver while the network driver software interface and the rerouting driver are executed at the same time;

at least one template, for receiving information from at least one template jump in the network driver software interface and comprising inserted code for controlling communications;

blocking code, for preventing processing of the patching code that is positioned after the blocking code and for allowing continuous operation of the network driver software interface; and

the inserted code, for replacing the blocking code and to evaluate[[ing]] rerouted information received by the template jumps.

**[The Remainder of this page has been left intentionally blank.]**

24. (Previously Presented) The rerouting driver of claim 23 wherein the control code identifies a starting memory address of the network driver interface instruction code and disables access to the first section of code, and further wherein the patching code, following the disabling of access with the blocking code, operates to overwrite the first section of code and additional pre-determined memory addresses so that all the pre-determined memory addresses are patched.

25. (Previously Presented) The rerouting driver of claim 23 wherein the patching code responsive to receipt of information being sent from the network driver software interface, determines the instruction code address that sent the information and overwrites the first section of code at that address so that memory addresses are incrementally patched as information is received from the network driver interface.

**[The Remainder of this page has been left intentionally blank.]**

26. (Currently Amended) A method for disabling and re-enabling access to code in a multiprocessor system having a shared memory and a network driver software interface comprising:

selecting a first section of code of the network driver software interface in a central processing unit that is to be modified while the network driver software interface is running;

writing the first section of code of the network driver software interface into the cache memory of the central processing unit while the network driver software interface is running;

overwriting a portion of the first section of code in cache memory with blocking code comprising code that causes a loop around serialization instruction in order to create a first version of code while the network driver software interface is running;

writing the first version of code into shared memory while the network driver software interface is running;

modifying the first version of code in the cache memory to create a second version of code, wherein a portion of the code following the blocking code is overwritten with template jumps for forwarding calls to a template to effect a static patch of the network driver software interface when the network driver software interface is running in the shared memory, the template comprising new computer code for controlling communications, the template jump and template allowing the new computer code to control communication between one or more media access control units and one or more protocol drivers;

writing the second version of code into shared memory while the network driver software interface is running;

modifying the second version of code in the cache memory with code to create a third version of code, wherein the blocking code is overwritten to remove the blocking code while the network driver software interface is running; and

writing the third version of code into shared memory while the network driver software interface is running.

27. (Cancelled).

28. (Currently Amended) A machine-readable medium having stored therein instructions, which when executed, cause a set of one or more processors to perform the following:

disabling access to a first section of code of a network driver software interface while the network driver software interface is running by overwriting code that is positioned before the first section of code with blocking code, the first section of code providing a communication path between a media access control unit and an application, the first section of code including a generic call;

overwriting the first section of code with a second section of code while the network driver software interface is running the blocking code;

overwriting the blocking code with the second section of code and without stopping complete operation of the network driver software interface;

executing the second section of code to cause execution flow to be rerouted to a third section of code in a rerouting driver, the second section of code being no larger than the first section of code,

the third section of code, when executed and while the network driver software interface is running the second section of code, completing the communication path and returning execution flow, the third section of code including additional code not present in the first section of code that is now inserted into the communication path and that controls communications between the media access control unit and an application.

29. (Original) The machine-readable medium of claim 28 wherein the second section of code contains a template jump to a template in the third section of code.

30-58. (Cancelled).

**[The Remainder of this page has been left intentionally blank.]**